Exercise 02 a)

**Design Report**

**Description:**

Changing outdated simple login system allowing bank customer to log into the system and check their current balance. But because system is outdated and does not provide much response to the user’s current errors or invalid entries are not displayed to the users. A project has been commissioned to further this application to better inform the user about what is happening within the system.

The object we will be using are specified bellow and in Appendix A:

Class Customer:

* First Name
* Last Name
* User Id
* Password
* Class Role
* Account Number
* Email Address

Class Role:

* Administration Full Access
* Administration Report Privileges
* Generate Audit Records
* View Audit Records
* Input Account Payments
* Authorise Account Payments
* Manage Account
* View Account Information
* View Account Balances

Class Help:

* Help text

**Appendix A: Password File Record Specification**

**Filename: p\_words.csv**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sequence** | **Column** | **Required** | **Data Type** | **Min Chars** | **Max Chars** | **Min Value** | **Max Value** | **Default Value** |
| 1 | First Name | Yes | String (Alpha) | 3 | 25 |  |  |  |
| 2 | Last Name | Yes | String (Alpha) | 3 | 25 |  |  |  |
| 3 | User Id | Yes | String (Alpha Numeric) | 8 | 8 |  |  |  |
| 4 | Password | Yes | String (Alpha Numeric) | 8 | 16 |  |  |  |
| 5 | Role | Yes | String (Alpha) | 3 | 35 |  |  | Customer |
| 6 | Account Number | Yes, for Role: Customer No for other roles | String (Numeric) | 9 | 9 |  |  |  |
| 7 | Email Address | Yes | String (Alpha Numeric) | 15 | 35 |  |  |  |
| 8 | Administration Full Access | Yes | Boolean | N/A | N/A | No | Yes | No |
| 9 | Administration Report Privileges | Yes | Boolean | N/A | N/A | No | Yes | No |
| 10 | Generate Audit Records | Yes | Boolean | N/A | N/A | No | Yes | No |
| 11 | View Audit Records | Yes | Boolean | N/A | N/A | No | Yes | No |
| 12 | Input Account Payments | Yes | Boolean | N/A | N/A | No | Yes | No |
| 13 | Authorise Account Payments | Yes | Boolean | N/A | N/A | No | Yes | No |
| 14 | Manage Account | Yes | Boolean | N/A | N/A | No | Yes | No |
| 15 | View Account Information | Yes | Boolean | N/A | N/A | No | Yes | No |
| 16 | View Account Balances | Yes | Boolean | N/A | N/A | No | Yes | No |

Exercise 02 b)

**Application**

**Exercise 2: Login System**

A bank has been developing a simple login system allowing their customer to log into the system and check their current balance. This system however is outdated and does not provide much response to the user’s current errors or invalid entries. A project has been commissioned to further this application to better inform the user.

This exercise consists of two parts; a report and an application.

**Exercise 2 a: Report**

Prepare a design specification report describing a solution for the above scenario. In your report identify the objects, data and file structures required.

Also include in your specification:

* Clear and concise technical and end user help documentation
* Discussion on how to load test the login system (resources, procedures etc.)

The report is to be between 750 and 1,000 words.

**Exercise 2 b: Application**

Based on the requirements of the report you developed in Exercise 2 a, implement the Login System in C# using Microsoft Visual Studio Integrated Development Environment.

A quick textual analysis of the scenario above yields the following key words:

* Bank
* Simple
* Login System
* Customer
* Errors and invalid entries
* Inform the user

Adding to the key words:

* Bank
* Think in terms of a general business as there are no “Bank” specific requirements.
* Simple
* Use role based permissions. Privileges are granted to the user based upon the user’s role within the bank. A user privilege is a right to execute a particular option within the application. See Appendix A
* Find a balance. The system needs to work for the bank and be generic enough to be easily applied to other systems and organisations
* It would be nice to incorporate a database into this application. However, at this stage of development, and to keep it simple – use a CSV file to store the login details (read only). For the purposes of this exercise use Excel or Notepad to write to the CSV file. See Appendix A for the structure of the CSV file
* Login System
* Login Credentials to this system are to consist of a User ID and Password
* User ID (label and text box)
* Password (label and text box) See Appendix A for details of the password strength requirements A C# class for validating passwords will be supplied to you
* Login (button) Greyed out until User Name and Password have been entered Check that the credentials entered are valid and if they are, close the login dialog and open the main application form
* Register (button) Open a dialog to register an existing customer for a new password, customer display the details, and solicit a password and simulate th e saving of the data to the CSV file
* Forgot Password (button) Open a dialog and solicit authentication details from the user Authenticate the user and email the user a new password (just simulate the email operation)
* Help (button) Display a help dialog including end user help text outlining how the login system works
* Customer
* The word customer is very specific, think in terms of end user. See Appendix A for list of end user roles for this system
* Errors and invalid entries
* Error messages Displayed via a message dialog
* Inform the user
* Display on screen help and “how to” details in a dialog

Notes:

* The focus of this exercise is for you to demonstrate your knowledge of event driven programming.
* A C# class for validating passwords will be supplied to you.
* The password CSV file; p\_words.csv will be supplied to you.
* Cater for a backdoor password. A backdoor password is a master password that grants the software developers access to the system without the need to create an account. For obvious reasons this password must be kept completely secret. The system will be compromised if it falls into the wrong hands. Use “fred” for both the backdoor User Id and Password, and grant full access privileges.
* Create a very basic main application form, just include a menu bar with a File menu that has an Exit option (which exits the application when selected).
* Your application must include a bare minimum of five of the following (you will probably need more than five).

Feel free to add to this list:

 Forms  Project  Common Dialogs  Classes  Dialogs  Public  Text Boxes  Private  Labels  Listeners  Command Buttons  Exceptions  Combo boxes  Variables  List boxes  Strings  Check boxes  Arrays  Radio buttons  Sequential File Handling

* Your application must:
* Compile
* Run
* Be fully compliant with the specification
* Include error message dialogs
* Include end user help and guidance